

CCC	000	M M	M M	000	DD	000	RRR	EEE
C	0 0	MMM	MMM	0 0	D D	0 0	R R	E
C	0 0	M M	M M	0 0	D D	0 0	RRR	EE
C	0 0	M,M	M M	0 0	D D	0 0	RR	E
CCC	000	M M	M M	000	DD	000	R R	EEE

```

      6  4
      6  4
    6 66 4 4
    6  6 4444
      66  4

```

FCV USER GROUP

 NEWSLETTER

JUNE 1984

No.3

So this is Newsletter No. 3 and just to keep things interesting we've got an exclusive from our technical expert on cracking program protection. But we'll save that for later - first a gripe. Where are those lists of programs etc that were requested last issue. The Software Sub-Committee needs you to respond, or else you won't get any nice, new games from(through) us. We can let you know of the person(s) most likely to swap with you to get the kind of program you are after, IF we have FEEDBACK.

The discussion at last Monday's meeting centred mainly around aspects of loading programs - some of which was beyond me so bad luck if you can't follow the following.

Two ways of loading programs from within other programs were described. The first should be used to load basic programs and is simply:-

```
20 load "xyz"
```

Note that "xyz" will automatically run once loaded and therefore should be the last statement accessed by the loader program. An example of this would be the first program sets up sprites and user defined character sets, and the second program actually runs the game.(Crazy Kong is an example of this). An alternative to be used when chain loading a machine language program is:-

```
10 a=a+1
20 if a=1 then load "xyz",1,1
30 if a=2 then sys 11111
```

In this case the loaded program will not execute until the appropriate SYS number is encountered. (Tech note: those among you with some programming knowledge may be wondering how the SYS is ever executed in the example above. What happens is that the program continues RE-running "forever". Therefore on the second run through, A picks up the value of 2, and the SYS is performed. You may ask how or why does this happen? Ans.- Bugged if I know!)

FCV USER GROUP

There was also a query about whether machine code can be got out of the computer without turning the machine off. The answer is to enter SYS 64738 to reset the pointers and create "power up" condition. This should make turning the machine off unnecessary.

The procedure for direct copying of tapes without using the computer should be of interest to tape users. A reasonably high quality cassette deck and a cheap tape recorder are required. The cheap tape recorder, which is plugged into the tape deck, is used to 'play' the tape. The cassette deck is used to record the signal on the blank cassette. The V. U. meter of the cassette deck should be kept in the red zone ie volume should be kept up very high. This procedure makes tape copying twice as fast as normal, allows protection to be copied and avoids tying up the computer.

(Tech note: This is a good way of copying tapes and bypassing protection, but recording level is extremely critical- just ask Ian Wild!)

Further to last week's message about subscribing to Compute's Gazette, Scott Shaw is keen to hear from anyone who would like to share the costs of subscription. He hopes to share the magazines around a group of about six people. Please contact Scott Shaw on extension 400.

CONVENOR'S CORNER

Keep those agenda items coming in! Page 5 of the New Leaf No 8 (May 18) raises an interesting point. Now that we have a flexible lunch time (from May 28) we have the possibility of longer meetings.

Is this what you want? If it is we may have to meet elsewhere to avoid conflict with staff taking shorter lunches. Anyway have a think about it. We'll discuss it on June 4 (not likely, since no-one showed up! We'll try again on the 18th) in Assessment Section starting at 12.45, preserving the status quo until the consultative process is through!

TECH TALK

by "the Ralph"

Topic : Program protection, and it's negation.

Programs are protected by the writer in order to stop unscrupulous people from reaping the rewards of the writer's effort, without rewarding the writer for those efforts. I will attempt to illuminate some of these methods, and try to give ideas on their negation for the eager and/or inexperienced.

The term negation is used to indicate that protection "cracking" is sometimes a subtle act. Protection can be altered, bypassed, or just plain removed.

Many people find that cracking programs is a most pleasurable activity. Matching wits with the programmer provides good mental exercise and stimulus. Actually, I believe that it is one of the best ways to improve your knowledge of your machine, because in order to alter some program you need to understand what the program code is doing at the particular point in question.

Now on with the show! One of the commonest and simplest forms of program protection is the "memory check/reset" method. This involves the program checking a particular memory location for a specific value. If an incorrect value is stored there, the code sends control to the "cold start" routine in memory, which has the same effect as turning your machine off!(well, nearly)

Fortunately this type of protection is easily seen (the screen shrinks horizontally and the power up message is displayed). There are 2 fairly easy ways to remove this barrier to your computing enjoyment, via either machine code or basic.

(1) Machine code.

After firing up your monitor, fill the basic memory with zeroes

```
F 0000 9FFF 00
```

Then load in the program

```
L 0001 "[prog.name]" 01
```

Finally, hunt for the machine code instruction JMP \$FCE2 by entering

```
H 0000 9FFF 4C E2 FC
```

Hopefully a 4 digit hexadecimal number will be output to the screen. If not, you are, shall we say, in the poo.

Now disassemble memory in the region of the number on the screen. E.g. if the number 090F was output earlier, then disassemble from about 0900 onwards.

```
D 0900 0920
```

This should give you a display similar to this :-

```
0900 LDA #$FE
0902 LDX #01
0904 STA $0000
0907 LDA $0348      ; checking the cassette buffer after a load
090A CMP #2A        ; is it a "*"
090C BEQ $0912      ; if it is then continue with the program
090F JMP $FCE2      ; if not bomb the program out
0912 [more instructions follow]
```

We can see the problem, and can therefore take appropriate action. We now have several options. We can alter the tape header to include the missing "*" by resaving the program with the "*" in the name at the right spot (but count carefully!), or we can alter the program so that it ignores the jump to cold start.

I have found that this is usually the easiest, so I will show how to do it.

```
Type A 090F NOP and return
Now type NOP return twice
Now type return
```

You have now successfully removed the protection from this program! All that is required now is to save the program to tape, the method of which I will leave to the inquisitive!

(2) BASIC

The methodology here is similar to that of the machine language programmer, but done via interpretive basic commands.

Firstly, load the program in.

Now, the tricky bit. We must search basic memory for the jump to cold start, so that we can nullify it. This is done by typing NEW and return. A short basic program is now typed in.

```

                                0 for I = 2049 to 16000:if
(peek(I)<>76) then next
  1 if ((peek(I+1)<>226)or(peek(I+2)<>252)) then next
  2 print "found at :";I:next

```

This checks for the jump code. If a "found at" message doesn't occur, then try changing line 0 to "16000 to 40000".

If a found at does occur, then write down then figure after the colon. Reload the original program and type in

```

    poke [found at],234
    poke [found at+1],234
    poke [found at+2],234

```

Now try to save the altered program, which will hopefully work.

CONGRATULATIONS, you are now a fully fledged "byte buster"! A few of the programs that the technique will work on are:

Centropods
Pakacuda

Hustler (although it doesn't use the cold start, but puts up a message saying "ILLEGAL COPY")

I think that I have taken up enough space this time, so until next issue, I bid you 'happy cracking and happy computing'.